



Chapter 4

Turing Machines

Limits of Computation

Clayton Johnson
Edna Reiter

TURING MACHINES



Alan Turing (British) 1912-1954

1936: *On Computable Numbers, with an application to the Entscheidungsproblem*

1939-1945: One of the British code-Breakers at Bletchley Park

1950: *Computing machinery and intelligence*



1951



Who was Alan Turing?

“Founder of computer science, mathematician, philosopher, codebreaker, strange visionary, and a gay man before his time”

<http://www.turing.org.uk/>



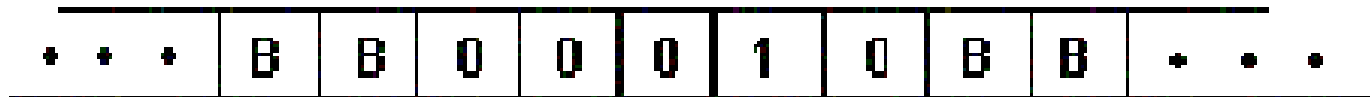
1946



TURING MACHINES: TAPE

- Tape: infinite in two directions
- Finite number of non-blank entries
- Σ input alphabet
- Γ tape alphabet
- blank (written **B**) member of Γ , not of Σ
- $\Sigma \subsetneq \Gamma$ since **B** $\notin \Sigma$

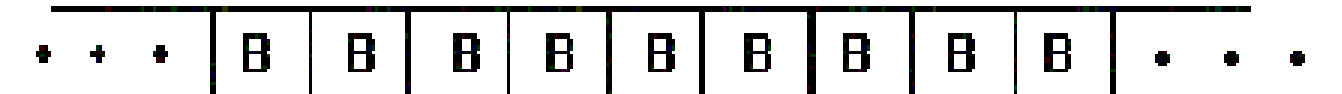
Turing Machine: tape examples



↑ ← Read-write head



↑ ← Read-write head



↑ ← Read-write head

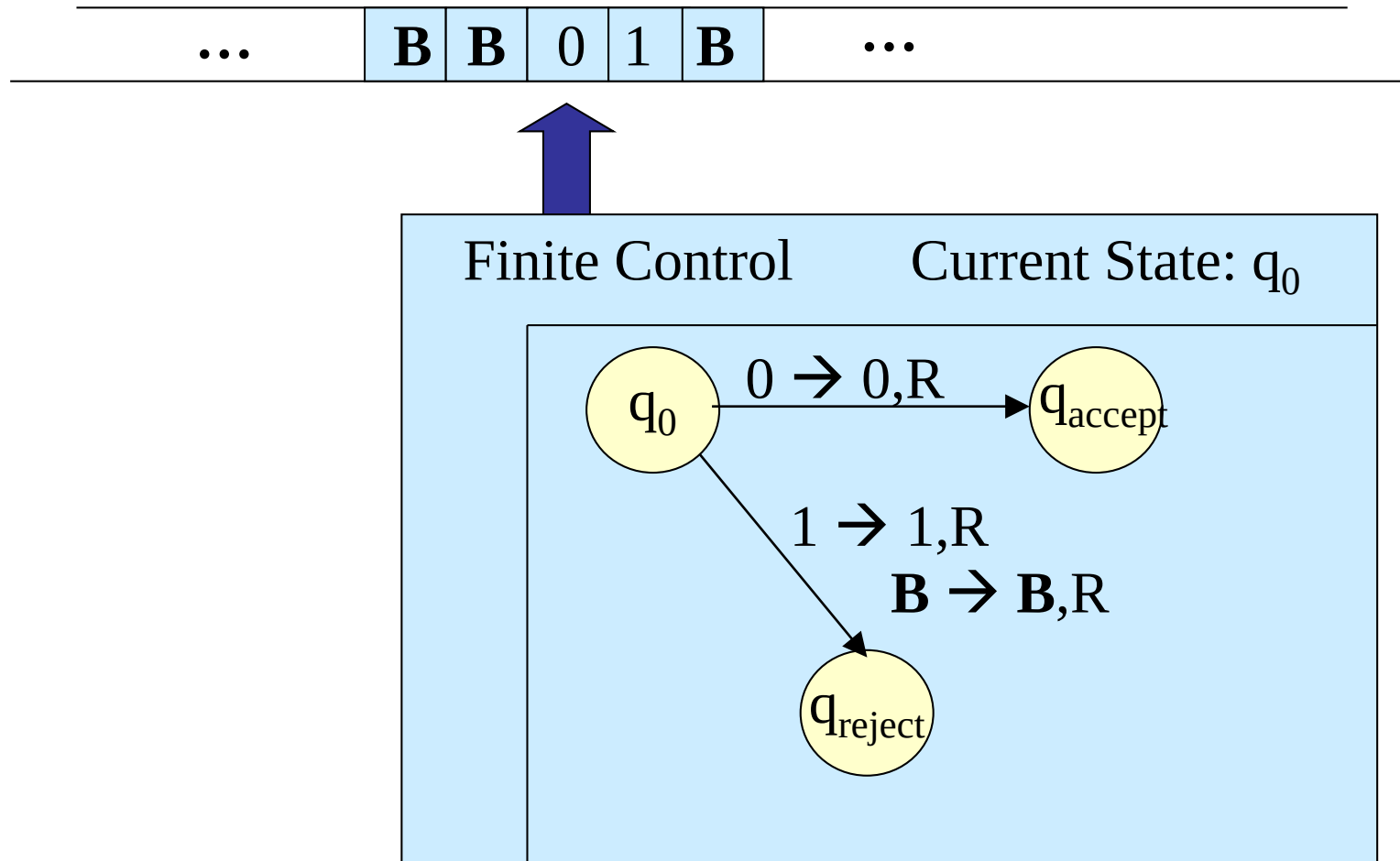
Read/write head: how does it work?



- Set of states
- What to do in each state
- q_{accept} q_{reject} : quit
- Other q_i : consult delta function δ
- Reading σ , write τ , move L/R, go to state q_j



Turing machine example





FORMAL DEFINITION

- A Turing machine is defined by three sets Σ , Γ , Q , a function δ , with specially designated members of Q for starting, accepting and rejecting, an infinite tape containing symbols, and a read/write head as follows:



DEF (con)

- An input alphabet Σ , consisting of a finite number of symbols, but not containing the blank symbol **B**.
- A tape alphabet Γ , also a finite set of symbols, containing at least the blank symbol (written **B**) and having Σ as a proper subset.



DEF (con)

A finite set of states Q , that contains at least a specially designated start state q_{start} and two distinct states q_{accept} and q_{reject} .

(Q must be disjoint from Γ , and must not contain either of the symbols “Left” or “Right”)

DEF (con)



A tape – or set of cells – that is infinite in two directions, both to the left and to the right. Each cell contains one symbol from the set Γ . The tape is assumed to start with only a finite number of non-blank symbols, and these non-blank symbols form a string containing no blanks.



DEF (con)

A transition function δ from $q' = Q - \{q_{\text{accept}}, q_{\text{reject}}\} \times \Gamma$ into the set of triples $Q \times \Gamma \times \{\text{Left}, \text{Right}\}$.

- ✓ for every state – except the accept and reject states – and every symbol, one and only one move is defined
- ✓ once the machine has reached an accept or reject state, it makes no more moves and remains in that state
- ✓ the machine is required to write a symbol at each move



DEF (con)

A read-write head that begins on the first non-blank input symbol (if there are any non blank input symbols).

The head will start on a blank **B** exactly when the input string is the empty string ϵ



TURING MACHINES TO:

- String contains substring 010
- String not contain substring 010
- 0 in the center of the string
- String is a palindrome
- String contains same number of 0's as 1's
- String contains binary nos. x, y, z with $z = x + y$
- String denotes prime number in binary

Some of these are really hard – but all can be done



TM Options

- Describe in English
 - Example “Find the right end of string”
 - Or “For each 0, cross it out and cross out a 1”
- Write out states and transitions
 - Show all the arrows
 - Can use conventions (arrows not shown reject)



Turing machines (Example M_1)

$M_1 = (\Sigma, \Gamma, Q, q_{\text{accept}}, q_{\text{reject}}, q_0, \delta)$ in which:

- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, \mathbf{B}\}$
- $Q = \{q_0, q_{\text{accept}}, q_{\text{reject}}\}$
- $\delta(q_0, 0) \rightarrow (q_{\text{accept}}, 0, \text{Right})$
- $\delta(q_0, 1) \rightarrow (q_{\text{reject}}, 1, \text{Right})$
- $\delta(q_0, \mathbf{B}) \rightarrow (q_{\text{reject}}, \mathbf{B}, \text{Right})$

What does this machine do?



Turing Machines

Examine transition function δ :

- $\delta(q_0, 0) \rightarrow (q_{\text{accept}}, 0, R)$

If first symbol is 0, accept

- $\delta(q_0, 1) \rightarrow (q_{\text{reject}}, 1, R)$

If first symbol is 1, reject

- $\delta(q_0, \mathbf{B}) \rightarrow (q_{\text{reject}}, \mathbf{B}, R)$

If first symbol is **B**, reject

What language is this?

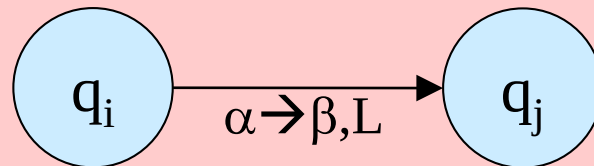
$L_1 = \{w \mid w \text{ begins with } 0\}$



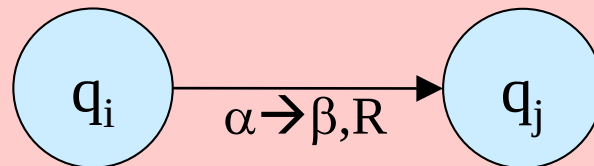
State Diagrams

A **state diagram** is a graphical depiction of a Turing machine:

Each $\delta(q_i, \alpha) \rightarrow (q_j, \beta, \text{Left})$



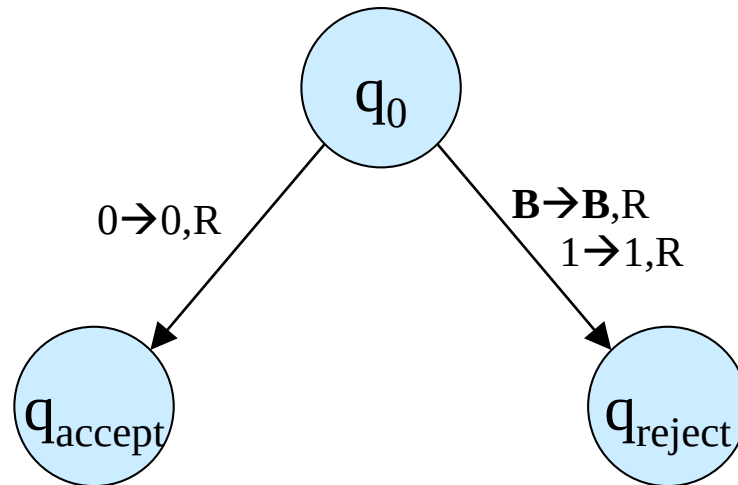
Each $\delta(q_i, \alpha) \rightarrow (q_j, \beta, \text{Right})$





State Diagram of M_1

$L_1 = \{w \mid w \text{ begins with } 0\}$



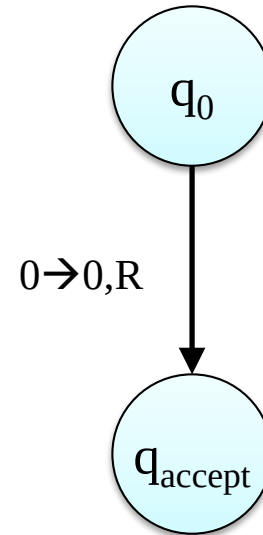
By convention, we don't need to show transitions to q_{reject} of the form

$$\delta(q_i, \sigma) \rightarrow (q_{\text{reject}}, \sigma, R)$$



Another State Diagram of M_1

$L_1 = \{w \mid w \text{ begins with } 0\}$



By convention, we don't need to show transitions to q_{reject} of the form

$$\delta(q_i, \sigma) \rightarrow (q_{\text{reject}}, \sigma, R)$$



Configurations

A **configuration** is a complete description of a Turing machine at some step of its operation.

Example:

$q_{\text{current}}; \underline{0}1011101$

It specifies three things:

- the current state of the machine**
- the entire non-blank portion of the tape**
- the location of the read/write head**



Specific Configurations

Starting configuration (example):

$$q_{\text{start}}; \underline{0}1101$$

Accepting configuration (example):

$$q_{\text{accept}}; 001\mathbf{B}\underline{1}101$$

Rejecting configuration (example):

$$q_{\text{reject}}; 0\underline{1}01\mathbf{B}101$$

TWO WAYS



- UNDERLINE HEAD POSITION

$q_{342}, 01\underline{1}00$

- PUT HEAD INTO THE TAPE STRING

$01q_{342}100$

Different texts will use different conventions



Computation Histories

A **computation history** is the sequence of configurations that a Turing machine enters when run on input w .

Machine M_1 has only three possible computation histories on input $w = w_1w_2\dots w_n$:

History 1

$q_{\text{start}}; \underline{0}w_2\dots w_n$

$q_{\text{accept}}; 0\underline{w}_2\dots w_n$

History 2

$q_{\text{start}}; \underline{B}$

$q_{\text{reject}}; \underline{BB}$

History 3

$q_{\text{start}}; \underline{1}w_2\dots w_n$

$q_{\text{reject}}; 1\underline{w}_2\dots w_n$

DEFINITION 4.3



A Turing machine M *accepts* (or *recognizes*) language L over Σ if for any string w in Σ^* , M enters q_{accept} if and only if w is in L . The language L is *recursively enumerable (R.E.)* (accepted by a Turing machine M).

Definitions



- **Definition 4.5:** A Turing machine M *halts* an input string $w \in \Sigma^*$ iff it enters either q_{accept} or q_{reject} .
- **Definition 4.6:** A Turing machine M *loops* an input string w iff it *never* enters either q_{accept} or q_{reject} .



DEFINITION 4.7

A Turing machine M ***decides*** language L over Σ if for any string w in Σ^* , M enters q_{accept} if w is in L , and enters q_{reject} if w is not in L . The language L is ***recursive*** (decided by a Turing machine M).



COMPARISON

M *decides* language L over Σ if for any string w in Σ^* , M enters q_{accept} if w is in L, and enters q_{reject} if w is not in L

M *accepts* (or *recognizes*) language L over Σ if for any string w in Σ^* , M enters q_{accept} if and only if w is in L.).



Definitions (redux)

A machine M ***accepts*** string w if it has an accepting computation history.

A machine M ***rejects*** string w if it has a rejecting computation history.

A machine M ***halts*** on string w if it has either an accepting or rejecting computation history.

A machine M ***loops*** on string w if it has neither an accepting or rejecting computation history.

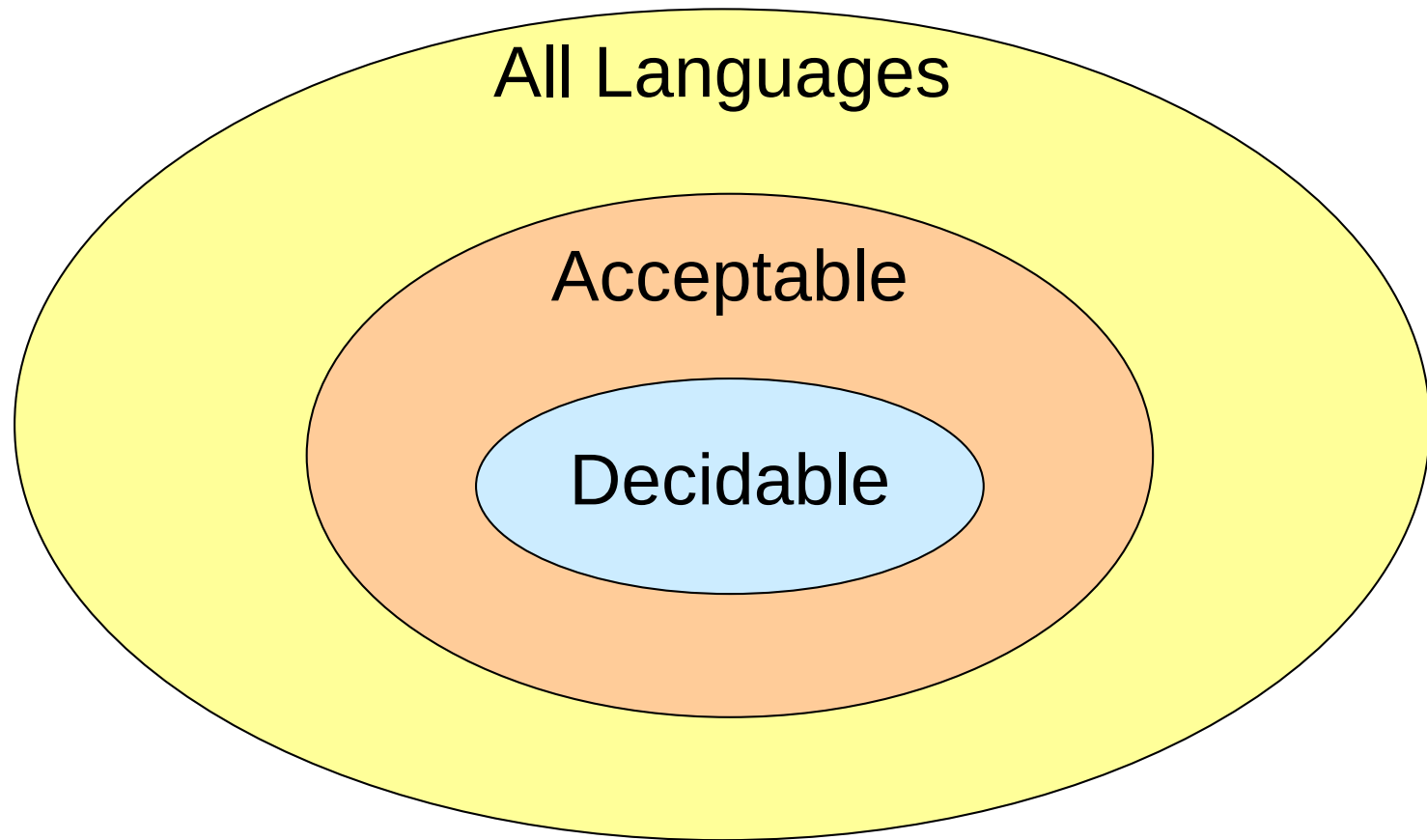
Decide/Accept (written with computational histories)



A Turing machine M ***decides*** a language L iff every input string $w \in L$ has an accepting computation history and every $w \notin L$ has a rejecting computation history.

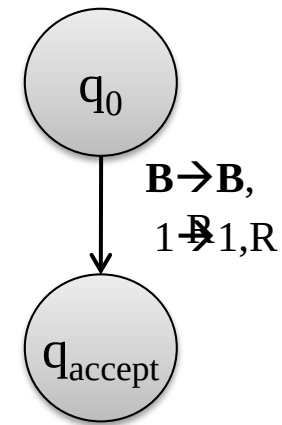
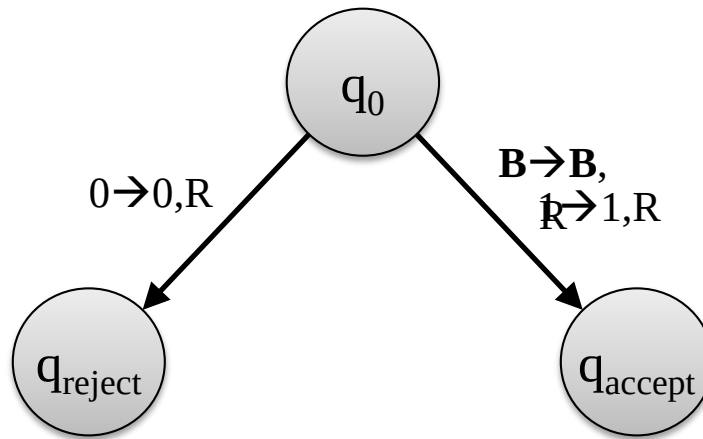
A Turing machine M ***accepts*** or ***recognizes*** a language L iff every input string $w \in L$ has an accepting computation history.

Languages



Turing Machines (example M_2)

M_2 decides $\{w \mid w \text{ does not begin with } 0\}$



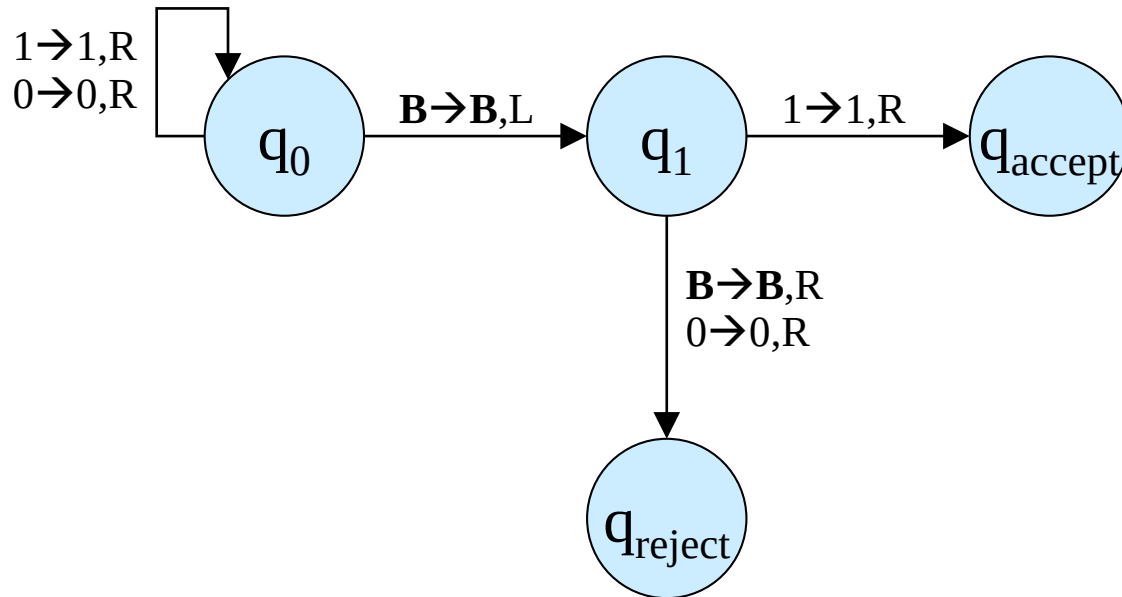
M_2 ACTION: look at first character
If not 0, accept else reject

Figure 4.4 Two pictures of the machine M_2

Turing Machines (example M_3)



M_3 decides $\{w \mid w \text{ ends in } 1\}$
(move till off rt end, move back, check char)

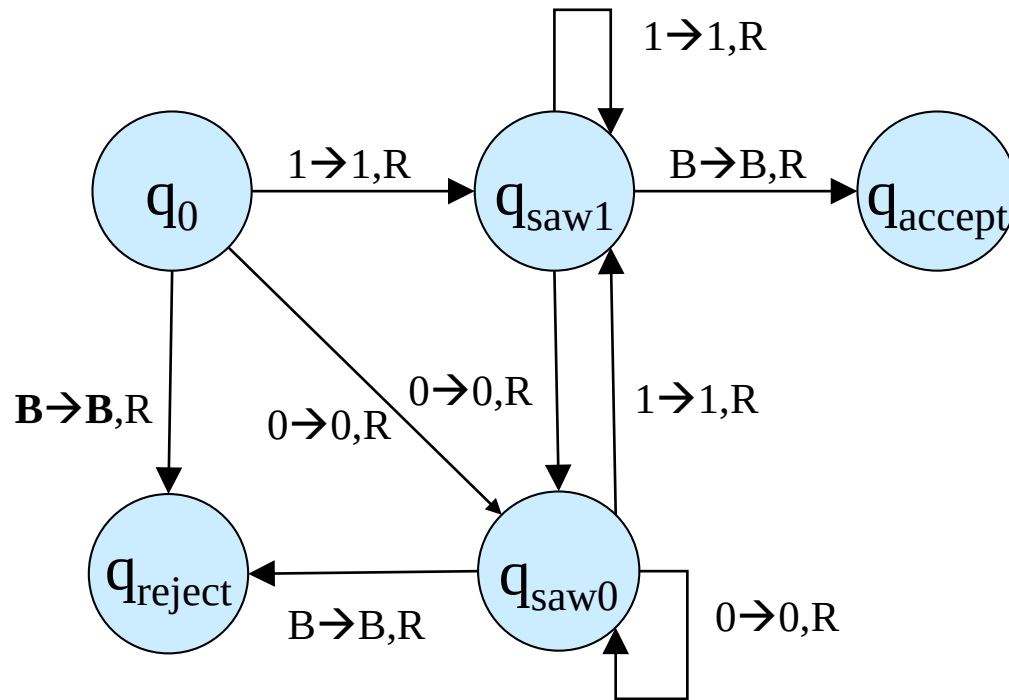


Is there another way to do this?

Turing Machines (example M_3)



M_3 decides $\{w \mid w \text{ ends in } 1\}$
(remember cur char, move rt, if hit end on 1 accept)

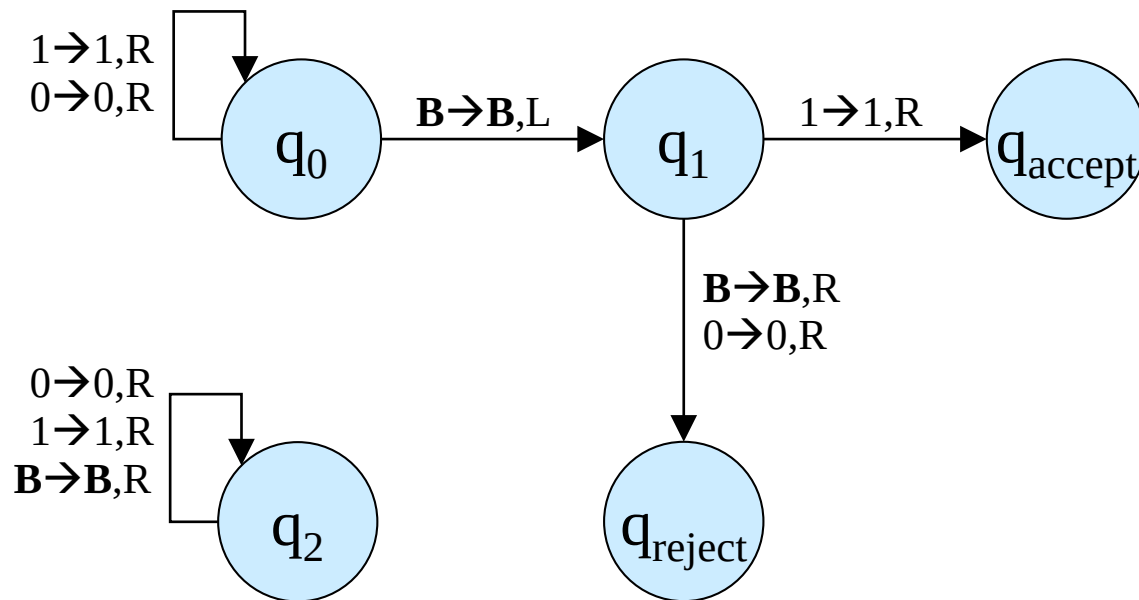


Turing Machines

(infinite alternatives)



You can always just add a **useless** state....

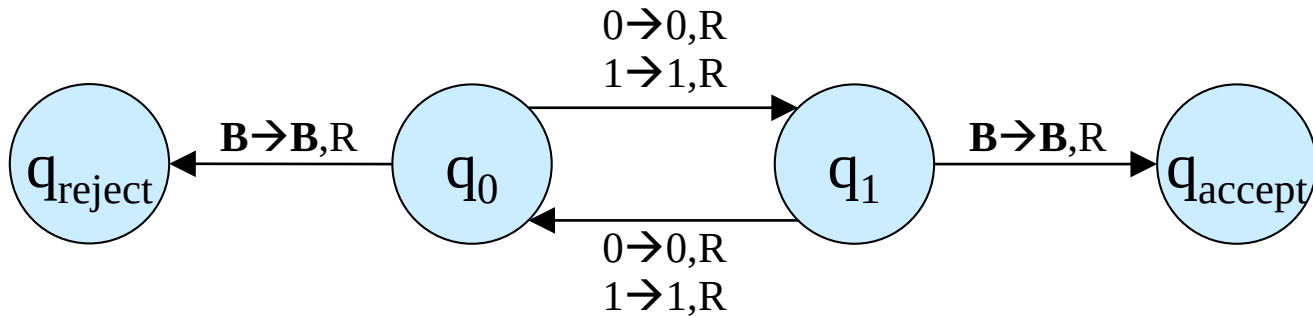


There are infinitely many equivalent machines



Turing Machines (example M_5)

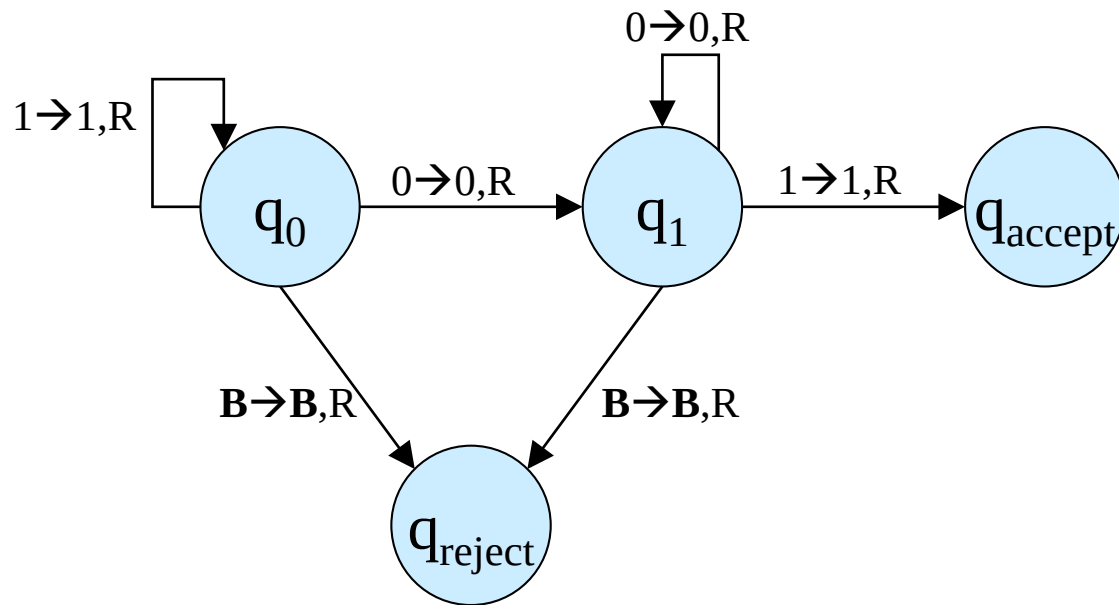
M_5 decides $\{w \mid w \text{ has an odd length}\}$





Turing Machines (example M_6)

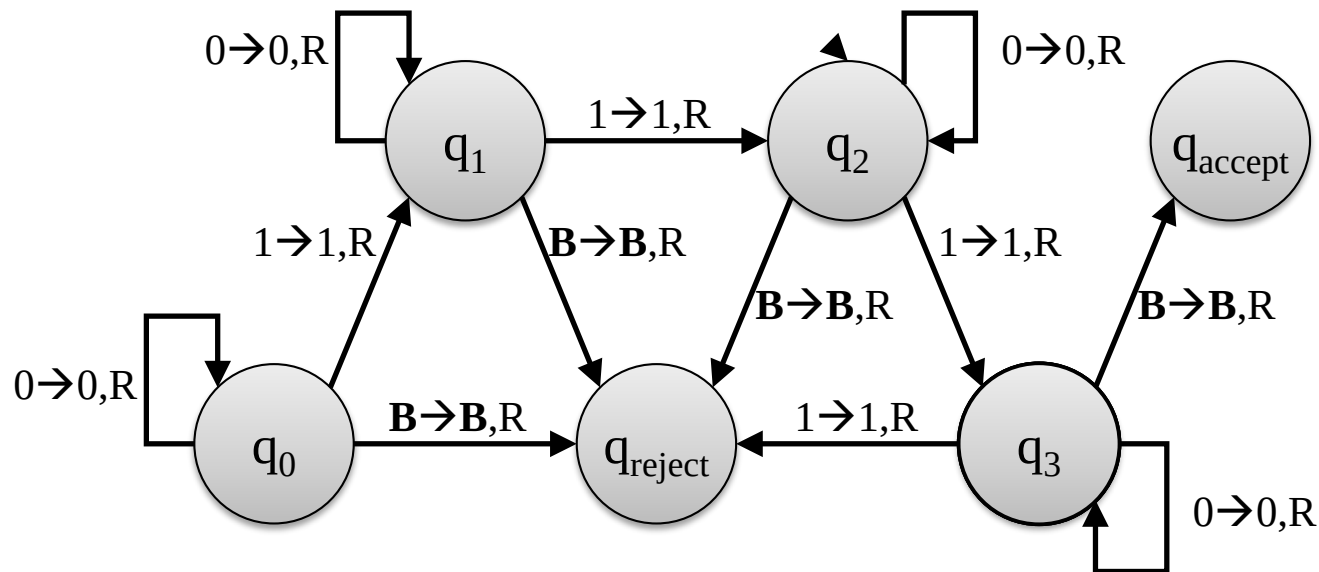
M_6 decides $\{w \mid w \text{ contains the substring } 01\}$





Turing Machines (example M_7)

M_7 decides $\{w \mid w \text{ contains exactly three 1's}\}$





Regular Languages

So far, our example Turing machines have not changed the contents of their tapes...

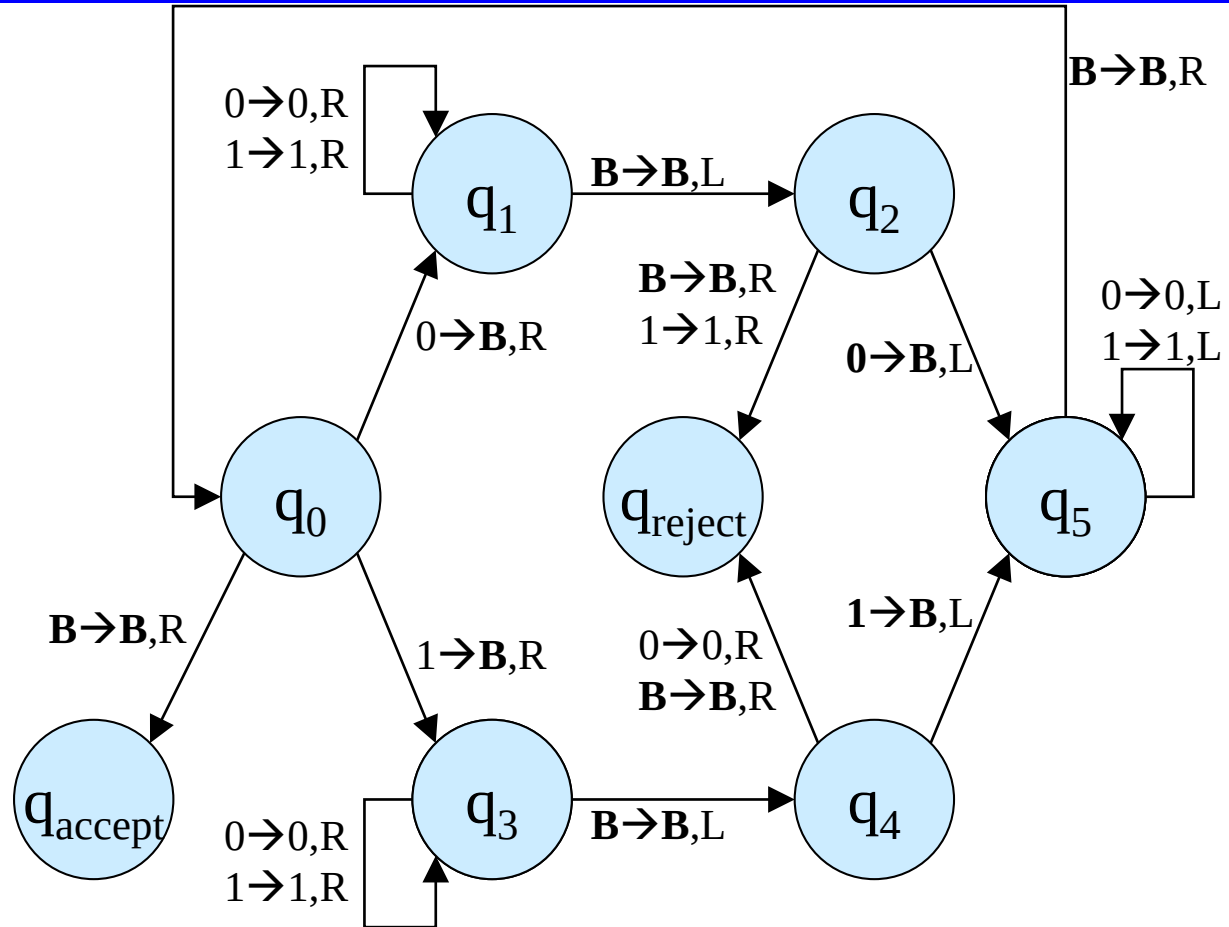
A language is ***regular*** if a Turing machine can decide it without altering its tape.



Turing Machines (example M_9)

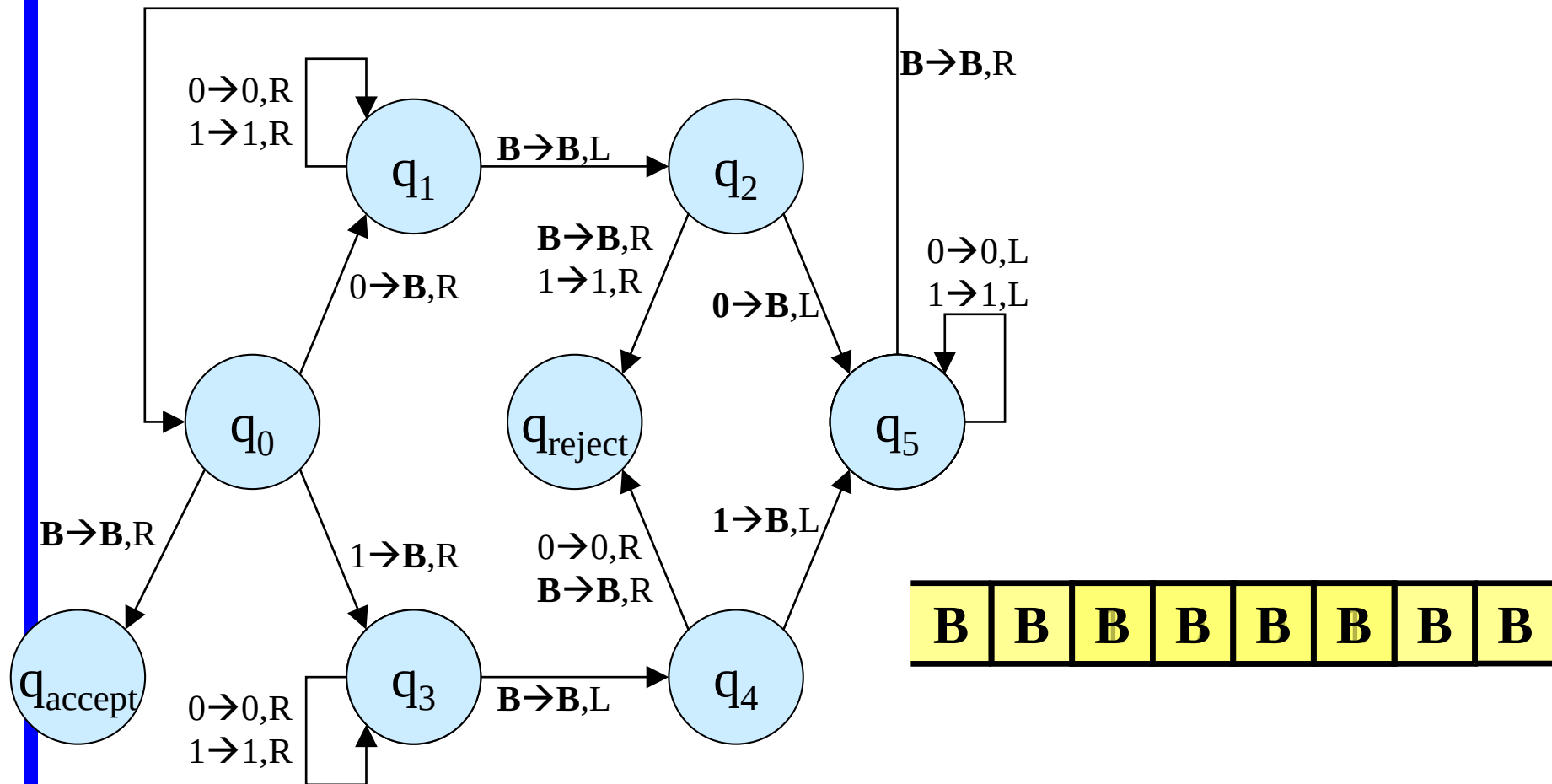
M_9 decides
 $\{w \mid w = xx^R\}$

Let's look at this
machine running
on input **1001**





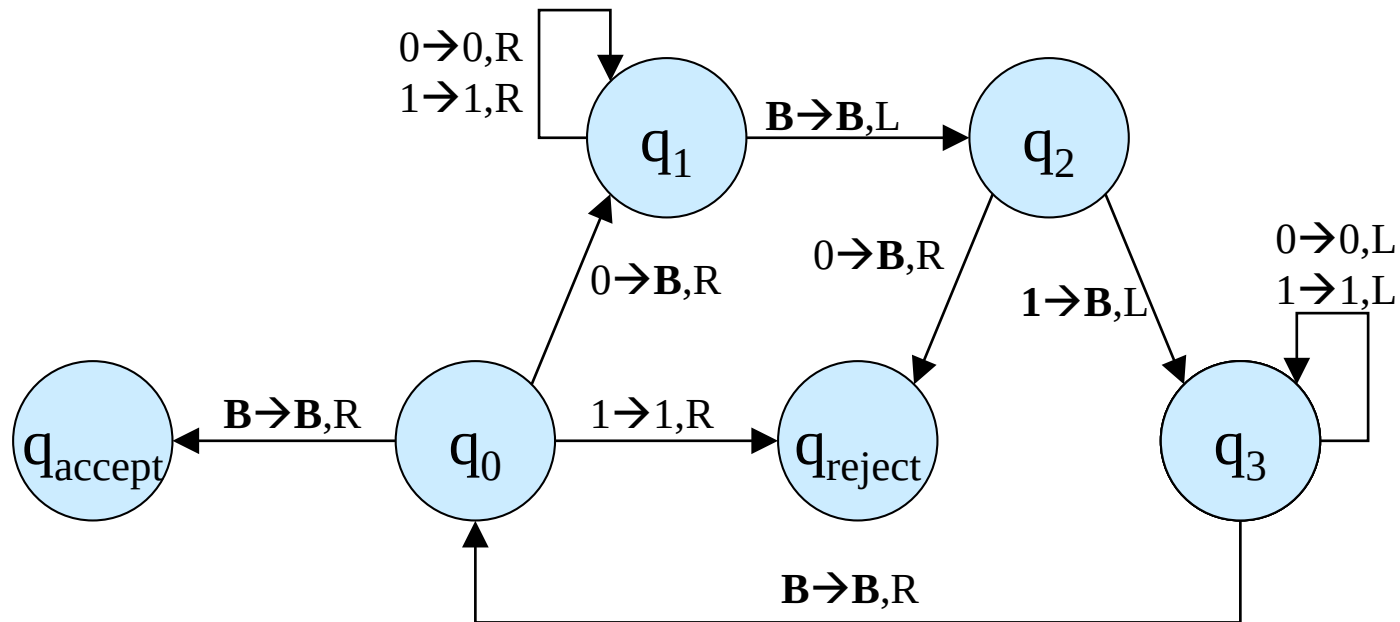
Turing Machines (example M_9)





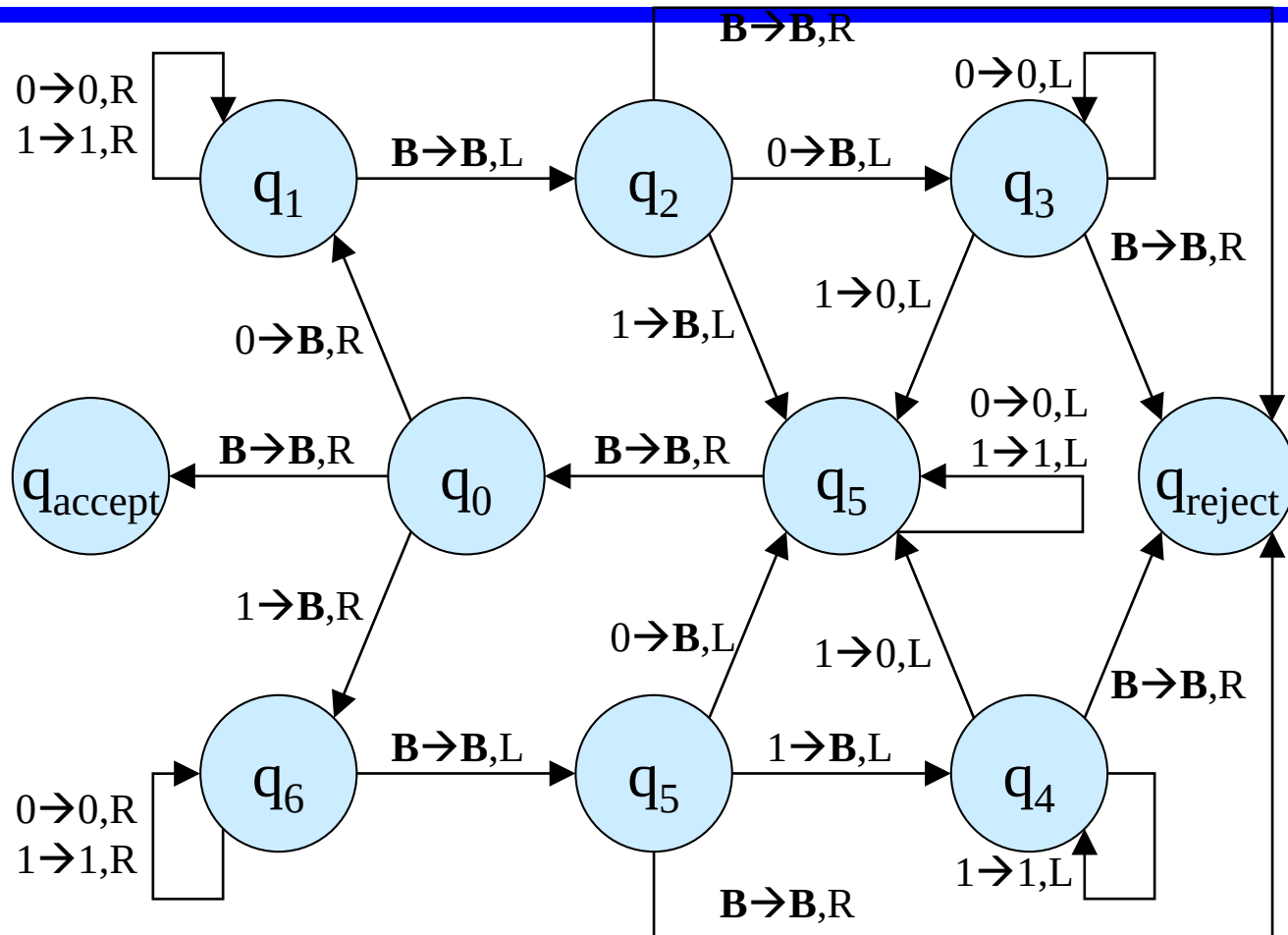
Turing Machines (example M_8)

M_8 decides $\{w \mid w = 0^n 1^n\}$





Turing Machines (example M_{11})



Is this a legal TM? What language is this?



Turing Machine Descriptions

State diagrams are cumbersome when describing complex Turing machines.

We can also offer an English language description of a machine's behavior.

Almost all Turing machines presented in this course will be discussed *algorithmically*.

Turing Machine Descriptions (example 1)



$M_1(w) =$

If first symbol is “0”, accept;
otherwise, reject.

What language is this?

$L_1 = \{w \mid w \text{ begins with } 0\}$

Turing Machine Descriptions (example 2)



$M_2(w) =$

Scan across w to find a blank.

Move left one cell.

If last symbol is a '1', accept;
otherwise, reject.

What language is this?

$L_2 = \{w \mid w \text{ ends in } 1\}$

Turing Machine Descriptions (example 3)



$M_3(w) =$

Let parity = even.

Repeat until a blank is found

If parity = even, let parity = odd;
otherwise, let parity = even.

Read next symbol to right.

If parity = odd, accept;
otherwise, reject.

What language is this?

$L_3 = \{w \mid w \text{ has an odd length}\}$



Subsumption

$M(w) =$

Run M_3 on input w .

If M_3 decides no, reject.

Scan across w until a “1” or blank is found.

If a “1” is found, accept; otherwise, reject.

What language is this?

$\{w \mid w \text{ has an odd length and } w \text{ contains } 1\}$



And finally...

If two languages L_1 and L_2 are decided (accepted) by Turing machines, then so are the languages:

$$L_1 \cup L_2$$

$$L_1 \cap L_2$$

$$L_1 L_2$$

$$L_1^*$$

$$L_1^c \text{ (decidable only)}$$

...what else???

Closure of Complement (decidable languages)



Let M_1 decide language L_1

Complement(w) =

Run M_1 on input w .

If M_1 decides yes, reject;
otherwise accept.

Closure of Union (decidable languages)



Let M_1 decide language L_1
Let M_2 decide language L_2

Union(w) =

Run M_1 on input w .

Run M_2 on input w .

If either M_1 or M_2 decides yes, accept;
otherwise, reject.

Also true for M_1 and M_2 TM-acceptable, but the proof is a bit more complex. Why? Problem related to next homework!

Closure of Intersection (decidable languages)



Let M_1 decide language L_1
Let M_2 decide language L_2

Intersection(w) =

Run M_1 on input w .

Run M_2 on input w .

If both M_1 and M_2 decide yes, accept;
otherwise, reject.

Closure of Concatenation (decidable languages)



Let M_1 decide language L_1
Let M_2 decide language L_2

Concatenation (w) =

For cut = 1 to $|w|$

Let $x = w_1 w_2 \dots w_{\text{cut}-1}$.

Let $y = w_{\text{cut}} w_{\text{cut}+1} \dots w_n$.

Run M_1 on input x .

Run M_2 on input y .

If both M_1 and M_2 decide yes, accept.

Reject.